(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 194 533**
A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 86102664.9

(22) Date of filing: 28.02.86

(51) Int. Cl.⁴: G06F 9/46

(30) Priority: 01.03.85 US 707304

(43) Date of publication of application:
17.09.86 Bulletin 86/38

(84) Designated Contracting States:
BE DE FR GB

(71) Applicant: WANG LABORATORIES INC.
One Industrial Avenue
Lowell, MA 01851(US)

(72) Inventor: Delyani, Dino
31 Appleton St.
Arlington, MA, 02174(US)
Inventor: Jablow, Charles E.
169 Highland Avenue
Sommerville, MA, 02143(US)

(74) Representative: Behrens, Dieter, Dr.-Ing. et al
Patentanwälte WUESTHOFF-V.
PECHMANN-BEHRENS-GOETZ Schweigerstrasse 2
D-8000 München 90(DE)

(54) Data processing system having tunable operating system means.

(57) A tunable operating system in a multiprogrammed data processing system (10) prevents lockout of I/O bound tasks of low priority by CPU bound tasks of high priority. A range signal in the master control block (28) and a range divider signal in each task control block (30) together define a queue subset (36, 42) for each task. The queue subsets (36, 42) for different tasks are overlapped by at least one queue, and preferably by at least three queues. A plurality of time-slice values, preferably two, are assigned when tasks are dispatched; the time-slice values are assigned with respect to the task range divider signal and therefore are not fixed with respect to each queue. The invention permits the weight given to the user-set task priority and to the priority based on recent task behaviour to be varied as desired. The range value, the divider values, and the time-slice values are variable to permit tuning of the operating system.

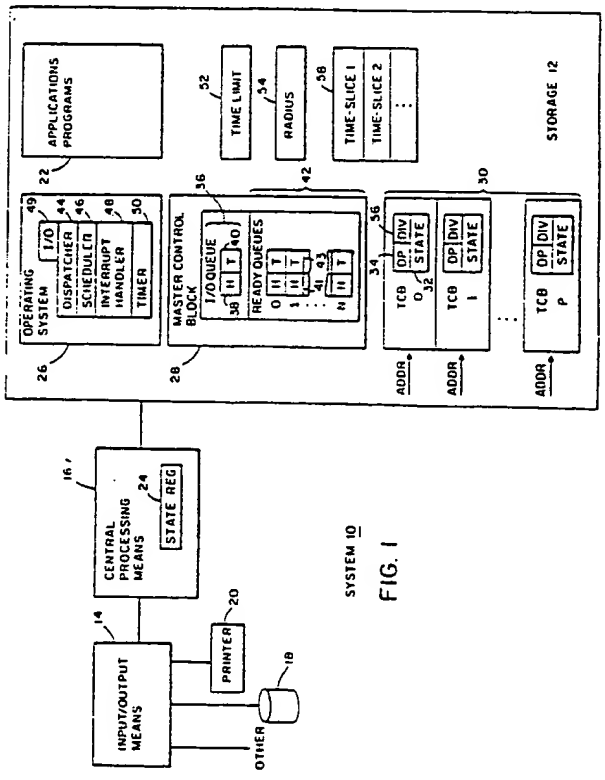

FIG. 1

EP 0 194 533 A2

According to the present invention, the data storage means further provides at least one radius storage element providing a queue radius signal, the queue radius signals together being representative of a total number of queues M less than N. Each task control block has a queue range divider storage element providing a queue range divider signal, the divider signal being derived from a user-set priority for the associated task.

The range divider signal and the range radius signals together define for each task a subset of queues divided into upper and lower ranges, and including highest and lowest priority queues for the task, the queue subset of any concurrent task having at least one queue in common with the queue subset of any other concurrent task, providing overlapped queue subsets.

For each task, the scheduler means is additionally responsive to the queue range divider signal and the queue radius signals for placing the associated task control block address signals in a particular ready queue within the queue subset defined for the task.

Further according to the present invention, the data storage means further provides a plurality of time-slice storage elements providing a plurality of discrete time-slice values, including a shortest and a longest time-slice value and monotonically increasing therebetween. For each task, the dispatcher means is responsive to the queue range divider signal and to the queue radius signal for providing a time interval limiting value signal responsive to the shortest time-slice value when the task control block address signals are retrieved from the highest priority queue in the subset, and providing a time interval limiting value signal responsive to the longest time-slice value when the task control block address signals are retrieved from the lowest priority queue in the subset, and providing time interval limiting value signals responsive to the monotonically increasing time-slice values corresponding to the remaining queues of the subset of monotonically decreasing priority.

In preferred embodiments, the data storage means further provides first and second time-slice value storage elements providing first and second time-slice signals, representing values of shorter and longer time-slices respectively. For each task, the dispatcher means is responsive to the queue range divider signal and to the queue radius signal for providing a time interval limiting value signal responsive to the first time-slice value signal when the task control block address signals are retrieved from a ready queue in the queue subset upper range, and providing a time interval limiting value signal responsive to the second time-slice value signal when the task control block address signals are retrieved from a ready queue in the queue subset lower range. The task range divider values and the radius values are related such that any task to be dispatched with the first time-slice value can be assigned to a queue having priority at least equal to that of the highest priority queue to which any other task to be dispatched with the second time-slice value can be assigned.

Preferably, the first time-slice signal represents a value of between 1/2 and 1 times the average CPU burst time of the I/O bound tasks concurrently running in the data processing system, and the second time-slice signal represents a value of between 1/2 and 1 times the average CPU burst time of the CPU bound tasks concurrently running in the data processing system.

Other objects, features and advantages will appear from the following description of a preferred embodiment of the invention, together with the drawing, in which :

Fig. 1 shows the data processing system in which the present invention is practiced, including the allocation of portions of storage;

Fig. 2 illustrates a particular implementation of the invention; and

Fig. 3 illustrates variations of the implementation of Fig. 2.

Referring now to the drawing, and in particular to Fig. 1, the present invention operates in a data processing system 10, having storage means 12, input/output means 14, and central processing means 16.

CPU. Central processing means 16 operates to control the operation of system 10. In particular, central processing means 16 is constructed and connected to control storage means 12 for placing signals therein representative of data or retrieving such signals therefrom. ("Data" is used herein to refer both to programs and to computational or other data operated on by such programs.) Central processing means 16 is further constructed and connected to control input/output means 14 to provide I/O functions. For example, devices such as disks 18, printers 20, or other devices such as a keyboard, a monitor, or a telecommunications link (not shown) are controlled by means 14.

Operating system. Storage means 12 provides means 26 for providing signals representing an operating system program. Further, storage means 12 provides storage elements at 28 for signals defining a master control block data structure associated with the processing means 16 operating under the control of the operating system program. (If system 10 has more than one processor, there will be more than one master control block.) Under control of the operating system program, with reference to signals stored at master control block 28, central processing means 16 controls means 12 and means 14 as previously described, and also provides other system services not pertinent herein.

Applications programs. Storage means 12 further provides means at 22 for providing signals representing various applications (user) programs. Central processing means 16, when controlled by such an applications program, executes the program in a manner well understood in the computer art, either by performing computational operations prescribed by the program or by requesting and performing operating system services such as a function of I/O means 14.

Tasks. A "task" is defined as a stream of work; successive programs may be executed, each successively identified with a common ongoing task. In a multiprogrammed data processing system, a plurality of concurrent tasks are present and ongoing in the system, each requiring a share of the services of central processing means 16. Such tasks may be applications (user) programs such as those at 22. A particular interactive user or a particular workstation may be regarded as a task, for which various programs are successively executed. A workstation is an interactive or "foreground" task; a program running in batch (noninteractive) mode is a "background" task.

Task control blocks. Storage 12 provides storage elements at 30 for signals representing task control block data structures. Each task control block data structure 0 through P is associated with one of a plurality P of concurrent tasks or tasks present in system 10. A task control block is initially provided at the time when a task is "created" or recognized as present by system 10. Each task control block data structure has an associated address value representing its location within storage means 12.

In alternative implementations, the dispatch priority may be reset in response to the I/O call rather than the I/O completion; this is not material to the present invention. CPU means 16 further provides a dispatch signal, which results in operation according to dispatcher 44 as has been described.

Time-slice runout.Timer 50 provides a timer interrupt signal in response to the completion of the time interval defined by limiting value signal 52. If no request for I/O (or other non-CPU) services is made before the completion of the time interval, the executing task is said to have "run out its time-slice"; it needs the services of central processing means 16 for a further time before it reaches the point of requiring an I/O or other service. At the completion of the time-slice, timer 50 provides a timer interrupt signal. Central processing means 16 then operates under the control of interrupt handler 48, and is responsive to the timer interrupt signal for discontinuing execution of the currently executing task. Means 16 retrieves the current state signals from register means 24 and places such signals into the state register means 32 of the task control block data structure 30 associated with the discontinued task, and provides a second interrupt handling means signal.

In response to such signal, central processing means 16 operates further according to scheduler 46, and resets dispatch priority signal at 34 in the task control block of the discontinued task to a value representing a lower priority (in general, the priority is decremented by one); central processing means 16 further places the address signals associated with the discontinued task at the tail 43 of a ready queue 42 corresponding to the new dispatch priority. As before, if the old priority value placed the task in the lowest priority queue, the priority value is reset to its previous (old) value. Means 16 under the control of scheduler 46 provides the dispatch signal, which results in operation according to dispatcher 44 as has been described.

I/O bound and CPU bound tasks.When a plurality of concurrent tasks of varous kinds are present in a data processing system, each task will, during any particular period of time which is long compared with an execution cycle time of the central processor, have a characteristic execution time for which it uses the central processing means before voluntarily giving up control in order to get an I/O or other non-CPU service (CPU burst time). Different tasks will have different characteristic CPU burst times. Thus there will be a distribution of CPU burst times, from a few milliseconds to as much as several hundred milliseconds. It is commonly said that at any particular time the concurrent tasks fall into two categories, referred to as "I/O bound" and "CPU bound", and corresponding generally to different values of the characteristic bpu burst times.

For a data processing system for which the present invention has been implemented, a task is described as "I/O bound" if its characteristic CPU burst time is about 15 ms. A task is described as "CPU bound" if its characteristic CPU burst time is about 60 ms. In general, these values depend on the particular data processing system.

Generally, apart from limitations imposed by the speed with which the system I/O devices operate, I/O bound tasks are considered to be limited in the speed with which they are carried out by the availability of I/O devices, that is, by the time they spend waiting in an I/O queue.

In contrast, apart from the speed of CPU operations, CPU bound tasks are considered to be limited in the speed with which they are carried out by the availability of the central processor, that is, by the time they spend waiting in a ready queue. Any particular applications program may at different times during its execution fall into first one and then the other of these categories.

Generally (and desirably for efficient use of the system components) a mix of tasks is present in a system at any particular time, some being I/O bound and some CPU bound. No rigorous definition of these categories is necessary.

User-set priorities.In many data processing systems, a task is assigned to a range of ready queues corresponding with a user-set priority, and fewer in number than the total number N of ready queues provided by system 10. The dispatch priority is repeatedly reset as described above responsive to the behaviour of the task (that is, whether or not it runs out its time-slice) but it cannot be reset to place the task in a queue outside its assigned range. According to the prior art, the ranges for particular user-set priorities are disjoint. For example, a range of ready queues 4 -8 might be assigned to applications tasks of a particular priority, and a range of queues 9 -13 to a second set of applications tasks. A task of the second priority cannot rise to any of queues 4 -8, nor can any task of the first priority fall to any of queues 9 -13. As has been described, whenever a task is present in any of queues 4 -8 it will be dispatched ahead of any task present in queues 9 -13. That means that if a task in one of ques 4 -8 is heavily CPU bound, it is repeatedly reassigned to one of these queues and repeatedly gains control of the CPU; it can "lock out" an I/O bound task in one of queues 9 -13, that is, prevent it from gaining control of the central processing means and thus significantly degrade its performance.

Further according to the prior art, if more than one time-slice value is employed, the values are permanently associated with particular ready queues.

It has been found that with the scheduling and dispatching method described above, it is possible for a highly CPU bound task having a high user-set priority to lock out a task having a lower user-set priority, even though the latter task may be highly I/O bound and may require only a portion of a single time-slice before voluntarily giving up control of the central processing means.

In particular, it has been found that foreground tasks (such as interactive tasks) which are heavily CPU bound severely limit the throughput of I/O bound background tasks (such as batch jobs). While this may be acceptable in some circumstances, in other circumstances it is unacceptable. The present invention provides means for preventing this undesirable mode of operation.

The present invention.According to the present invention, storage means 12 further provides a radius storage element 54, providing at least one queue radius signal. The queue radius signals may represent an equal number of queues. Alternatively they may represent an equal number of queues above and below a middle queue; in such case, the total number of queues is odd. Alternatively, there may be unequal numbers of queues above and below a dividing line.

Each task control block 30 has a queue range divider storage element 56 providing a queue range divider signal. The divider signal is derived from a user-set priority (commonly selected from very high, high, medium and low) for the task associated with the task control block. The divider signal may, however, also depend on other factors,

the two tasks must compete in a single queue, and the BG/L task must sometimes wait while the FG/VH task enjoys a 45-ms time-slice. It would however be desirable to avoid such competition if the particular system permits it, by providing greater overlap.

The effect of providing two time-slice values is that a task, during most of the time it is present in the system, will be scheduled in one of four disjoint regions within its queue subset: the top region (when the task is I/O bound); the region just above and including the divider queue (when the task is moderately I/O bound); the region just below the divider queue (when the task is moderately CPU bound); or the bottom region (when the task is CPU bound).

Because each of these regions, for each category of task, includes at least one ready queue other than those comprising the same region for other categories, tasks running particular programs with high priority will receive preferential treatment as compared with tasks running the same programs with low priority.

Further, since the I/O bound queues (the upper range) of even the lowest priority group (BG/L) are the same as or, desirably, higher than those of the CPU bound queues - (lower range) of the highest priority group (FG/H), CPU bound tasks will not prevent any I/O bound task from obtaining service from the central processing means. This maximizes system resource utilization and performance.

Two unequal time-slice values will provide a considerable advantage in the operation of the data processing system, as I/O bound tasks will be discriminated from CPU bound tasks. However, the advantage can be increased by optimizing the choice of time-slice values.

In preferred embodiments, a shorter time-slice value has been chosen which is between 1/2 and 1 x the average CPU burst time of I/O bound tasks. Consequently, on the average, an I/O bound task will run out its first time-slice without issuing a call for I/O services, with the result that its dispatch priority will be decremented and it will be assigned to a lower priority queue. However, on the average, the task will call for I/O services during its second time-slice, with the result that its dispatch priority will be incremented, or restored to the value of the dispatch priority before the first time-slice. Thus on the average an I/O bound task will remain in an initially assigned queue, so long as its behaviour does not alter significantly. From time to time the task will require three time-slices before issuing an I/O call, thus having its dispatch priority lowered twice, but this will be infrequent, and will be balanced by occasions when the task will issue I/O calls in successive time-slices, thus raising its dispatch priority again.

Analogous behavior can be expected for CPU bound tasks if the second time-slice value is chosen to be between 1/2 and 1 x the average CPU burst time of CPU bound tasks. It has been found, in fact, that most program behavior causes tasks to "jiggle" or "float" over a range of three queues for periods of time which are long compared with the time-slice; thus each of the four regions discussed above is preferably three queues in length. When the behavior of the task changes, the task moves with a generally consistent drift to one of the other regions. It has been found that tasks spend comparatively little time in queues intermediate these regions, if such are provided. Therefore, in preferred embodiments, four 3-queue regions, or 12 queues in all, comprise a queue subset. In the implementation shown in Fig. 2, 13 queues are provided because of an implementation detail in the particular system for which it is designed, requiring that there be an odd number of queues. It is found in practice that tasks are seldom assigned to queue 4.

The time-slice values represented at 58 and the divider values represented at 56 are parameters which may be easily changed to alter the performance of system 10. Thus the scheduler and dispatcher of operating system 26 are said to be "tunable". The effect of altering the size of the queue overlap by varying the divider values will now be discussed.

Referring again to Fig. 2, it will be observed that a line A may be drawn which separates the region with which 8-ms time-slices are associated from the region with which 45-ms time-slices are associated. Line A slants downward from the left at an acute angle to the horizontal. Referring now to Fig. 3, line A is again represented. Conceptually, in the region above and to the right of line A are I/O bound tasks, in all user-set priority categories, while below and to the left of line A are CPU bound tasks, in all user-set priorities. This case corresponds to a minimum overlap of 6 queues between queue subsets of highest and lowest priorities (an overlap of 11 queues between adjacent queue subsets), and has been noted, the most I/O bound task in the BG/L category may be queued with the least CPU bound task in the FG/VH category, in queue 11.

Consider now the case, as in the prior art previously discussed, in which no overlap is permitted among the queue subsets for each user-set priority category of tasks. In this case, all tasks in, for example, queues 4 -8 will be dispatched ahead of all tasks in queues 9 -13. An I/O bound task in the 9 -13 queue subset will rise, as its dispatch priority is repeatedly incremented, to queue 9; a CPU bound task in the 4 -8 queue subset will fall, as its dispatch priority is repeatedly decremented, to queue 9; nevertheless, the I/O bound task in queue 9, even though dispatched with a relatively short time-slice, can be locked out by the CPU bound task in queue 8, even though dispatched with a relatively long time-slice. This non-overlapped allocation of queues is represented in Fig. 3 by a vertical line B. While high priority tasks receive good service in this scheme, during times of heavy system use low priority tasks may be locked out.

If, in contrast, all tasks, regardless of user-set priorities, are allowed to move to any ready queue (that is, there are no queue subsets associated with user-set priorities) then all I/O bound tasks will rise to the upper queues (and be dispatched with a relatively short time-slice) while all CPU bound tasks will fall to the lower queues and be dispatched with a relatively long time-slice. In essence, all I/O bound tasks will run ahead of all CPU bound tasks. Such a situation is represented in Fig. 3 by a horizontal line C. This allocation of queues is also known in the prior art. The disadvantage of this arrangement is that foreground or other high user-set priority tasks receive no better service than background tasks; if system use is heavy, the response time of foreground (interactive) tasks may be seriously degraded while the system runs I/O bound background tasks.

In contrast with both these extremes of the prior art, represented by lines B and C, a data processing system operating according to the present invention provides a better balance between the priority categories, and provides increased system throughput. Such throughput can be considered quantitatively in terms, for example, of number of transactions per hour, or number of standard job completions per hour, or other appropriate benchmark, as is well understood in the art.

When the minimum queue overlap between queue subsets is increased, the line dividing the I/O bound region from the CPU bound region tilts toward the horizontal line C (which represents complete overlap) as seen at D; when

ing address signals associated with said task control block of said discontinued task at the tail of a said ready queue corresponding to said new dispatch priority value,

characterized in that

said data storage means (12) further provides at least one radius storage element (54) providing a queue radius signal, said queue radius signals together being representative of a total number of ready queues M less than N,

each said task control block (30) having a queue range divider storage element (56) providing a queue range divider signal, said divider signal being derived from a user-set priority for said associated task,

said range divider signal and said range radius signals together defining for each said task a subset of ready queues divider into upper and lower ranges, and including highest and lowest priority queues for the task, the queue subset of any concurrent task having at least one queue in common with the queue subset of any other concurrent task, providing overlapped queue subsets,

for each task, said scheduler means (46) being additionally responsive to said queue range divider signal and said queue radius signals for placing said associated task control block address signals in a particular ready queue within said queue subset defined for said task.

2. The system of claim 1, wherein said data storage means further providing first and second time-slice value storage elements (58) providing first and second time-slice signals, representing values of shorter and longer time-slices respectively,

for each task, said dispatcher means (44) being responsive to said queue range divider signal and to said queue radius signal for providing a time interval limiting value signal responsive to said first time-slice value signal when said task control block address signals are retrieved from a ready queue in said queue subset upper range, and providing a time interval limiting value signal responsive to said second time-slice value signal when said task control block address signals are retrieved from a ready queue in said queue subset lower range,

said task range divider values and said radius values being related such that any task to be dispatched with said first time-slice value can be assigned to a queue having priority at least equal to that of the highest priority queue to which any other task to be dispatched with said second time-slice value can be assigned.

3. The system of claim 1, wherein said data storage means further providing a plurality of time-slice storage elements (58) providing a plurality of discrete time-slice values, including a shortest and a longest time-slice value and monotonically increasing therebetween,

for each task, said dispatcher manes (44) being responsive to said queue range divider signal and to said queue radius signal for providing a time interval limiting value signal responsive to said shortest time-slice value when said task control block address signals are retrieved from said highest priority queue in said subset, and providing a time interval limiting value signal responsive to said longest time-slice value when said task control block address signals are retrieved from said lowest priority queue in said subset, and providing time interval limiting value signals responsive to said monotonically increasing time-slice values corresponding to the remaining queues of the subset of monotonically decreasing priority.

4.The system of claim 2, wherein said first time-slice signal representing a value of between 1/2 and 1 times the average CPU burst time of the I/O bound tasks concurrently running in said data processing system.

5. The system of claim 2 or 4, wherein said second time-slice signal representing a value of between 1/2 and 1 times th average CPU burst time of the CPU bound tasks concurrently running in said data processing system.

6. The system of any of claims 2, 4 or 5, wherein said task range divider values and said radius values being related such that any process to be dispatched with said first time-slice value can be assigned to a queue having priority at least two increments higher than that of the highest priority queue to which any other process to be dispatched with said second time-slice value can be assigned.

7. The system of any of claims 1 to 6, wherein said upper and lower queue ranges are equal to within one queue.

8. A multiprogrammed data processing system (10) for the execution of a plurality of concurrent tasks, having

timer means (50),

data storage means (12), and

central processing means (16) for controlling said data storage means, and providing state register means (24),

said data storage means having

means (26) for providing signals representing operating system means for the control of said data processing system,

storage elements (28) for signals defining a master control block data structure associated with operation of said central processing means under control of said operating system means, and

storage elements (30) for signals defining a plurality of task control block data structures,

each said task control block data structure being associated with one of a plurality of concurrent tasks in said data processing system, each task control block data structure having an associated address value representing its location within said data storage means, each task control block data structure providing storage elements (32) for state signals representing a current execution state of said associated task, and a priority storage element (34) for storing a dispatch priority signal corresponding to a dispatch priority value,

said master control block data structure providing a plurality of queue storage elements (42) for signals representing particular said address values, said storage elements identifying the head and tail of each of N successive chained ready queues representing tasks queued for control by said central processing means, each said ready queue having a

priority ready queues for that task, the ready queue subset of any concurrent task having at least one queue in common with the ready queue subset of any other concurrent task, providing overlapped ready queue subsets,

said computer further providing scheduler means (46) responsive for each task awaiting control of said central processor, to its said associated dispatch priority value signal and to its said queue range limit signals for placing a representation of said task in a particular ready queue within said ready queue subset defined for said task.

10. The computer of claim 9, further comprising time-slice means (58) for defining shorter and longer time slices,

said range limit signals comprising range radius signals - (54) and for each concurrent task, range divider signals - (56), said range divider signals and said range radius signals together defining upper and lower ranges for each ready queue subset,

for each said defined next task, said dispatcher means being responsive to its said queue range divider signal, its said associated dispatch priority value signal, and to said time-slice means for causing said timing means to define a time interval equal to a said shorter time slice when said task is dispatched from a ready queue in said queue subset upper range for said task, and for causing said timing means to define a time interval equal to a said longer time slice when said task is dispatched from a ready queue in said queue subset lower range for said task.

QUEUE	FG/VH	FG/H	FG/M	FG/L	BG/VH	BG/H	BG/M	BG/L
4	8							
5	8	8						
6	8	8	8					
7	8	8	8	8				
8	8	8	8	8	8			
9	8	8	8	8	8	8		
10	8	8	8	8	8	8	8	
11	45	8	8	8	8	8	8	8
12	45	45	8	8	8	8	8	8
13	45	45	45	8	8	8	8	8
14	45	45	45	45	8	8	8	8
15	45	45	45	45	45	8	8	8
16	45	45	45	45	45	45	8	8
17		45	45	45	45	45	45	8
18			45	45	45	45	45	45
19				45	45	45	45	45
20					45	45	45	45
21						45	45	45
22							45	45
23								45

A

FIG. 2

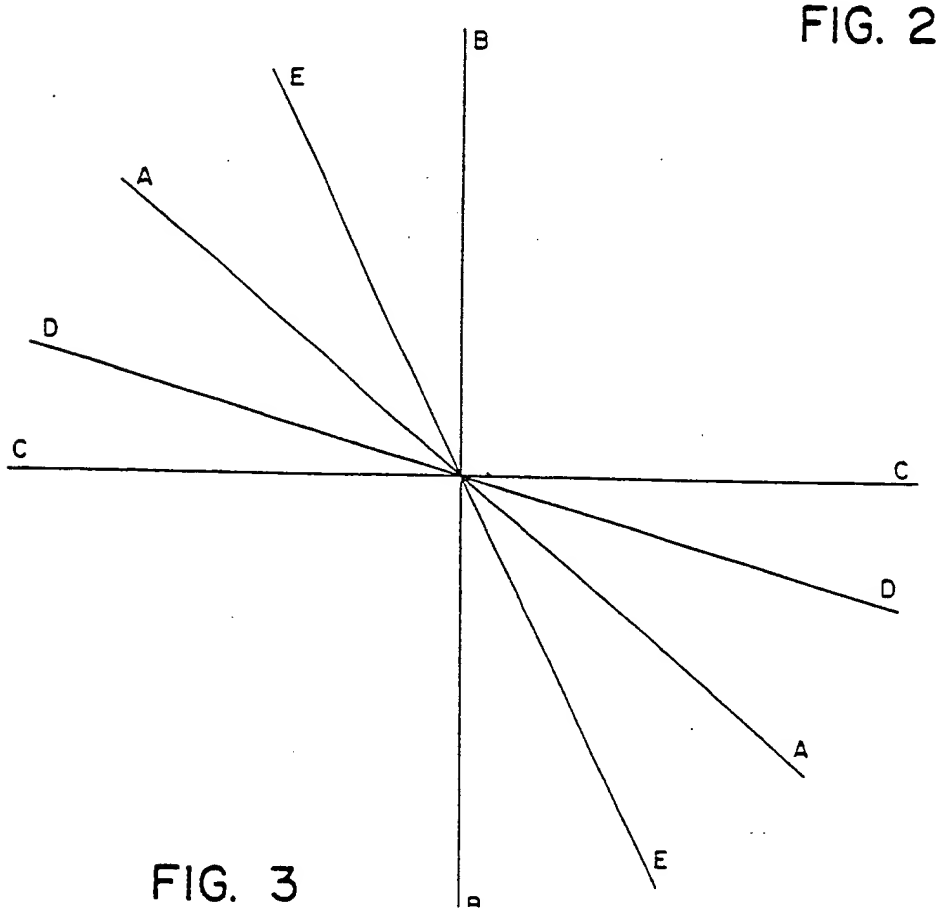

FIG. 3